2015-04-01

# Towards Rapid Recertification Using Formal Analysis

Smullen, Daniel; Breaux, Travis

Monterey, California. Naval Postgraduate School

SYM-AM-15-104


# PROCEEDINGS
## OF THE
# TWELFTH ANNUAL ACQUISITION RESEARCH SYMPOSIUM

## THURSDAY SESSIONS
## VOLUME II

## Towards Rapid Recertification Using Formal Analysis

Daniel Smullen, Carnegie Mellon University
Travis Breaux, Carnegie Mellon University

**Published April 30, 2015**




ACQUISITION RESEARCH PROGRAM
GRADUATE SCHOOL OF BUSINESS & PUBLIC POLICY
NAVAL POSTGRADUATE SCHOOL

The research presented in this report was supported by the Acquisition Research Program of the Graduate School of Business & Public Policy at the Naval Postgraduate School.

To request defense acquisition research, to become a research sponsor, or to print additional copies of reports, please contact any of the staff listed on the Acquisition Research Program website (www.acquisitionresearch.net).


ACQUISITION RESEARCH PROGRAM
GRADUATE SCHOOL OF BUSINESS & PUBLIC POLICY
NAVAL POSTGRADUATE SCHOOL

# Towards Rapid Recertification Using Formal Analysis

**Daniel Smullen—**is a research assistant enrolled in the software engineering PhD program at Carnegie Mellon University. His research interests include privacy, security, software architecture, and regulatory compliance. [dsmullen@cs.cmu.edu]

**Travis Breaux—**is an assistant professor of computer science in the Institute for Software Research at Carnegie Mellon University (CMU). His research program searches for new methods and tools for developing correct software specifications and ensuring that software systems conform to those specifications in a transparent, reliable, and trustworthy manner. This includes compliance with privacy and security regulations, standards, and policies. Dr. Breaux is the director of the CMU Requirements Engineering Lab, is a co-founder of the Requirements Engineering and Law Workshop, and has several publications in ACM- and IEEE-sponsored journals and conference proceedings. [breaux@cs.cmu.edu]

## Abstract

Department of Defense (DoD) acquisition requires IT to undergo the DoD information assurance certification and accreditation process (DIACAP), which makes architecture-dependent assumptions. Emerging IT architectures, such as mobile and cloud-based platforms, invalidate these assumptions and prevent the DoD from acquiring commercial technologies that are readily available to adversaries. To address this problem, we extended our initial automation framework, wherein an application profile is expressed in a formal language and scaled with evolving architectural assumptions. These profiles will help ensure that information assurance requirements are commensurate with risk and scalable based on an application's changing external dependencies. Information assurance risk levels must account for changing environmental and IA parameters (confidentiality, integrity, and availability) that result from dynamic recombination of applications during runtime. Our proposed language aims to address dynamically composable, multi-party systems that preserve security properties. Software developers and certification authorities can use these profiles expressed in first-order logic with an inference engine to advance the DIACAP and re-check compliance as IT systems evolve over time.

## Introduction

Ensuring confidentiality in information systems is of paramount importance to mitigate the likelihood of data spills. But as systems change and requirements evolve, it quickly becomes unclear how these changes may affect the security of protected information with regard to secure enclaves. The DoD is increasingly reliant on software in all operational contexts, and agility in terms of the ability to certify and deploy new and improved software technologies necessitates greater agility in certification processes. Software recertification processes require significant expenditure in order to provide evidence of information assurance (IA) policy conformance. The costs of both sourcing and developing software are compounded by the need to maintain these certifications, especially when changes occur. Current processes are manual, and cannot scale as complexity increases. Savings and scalability can be achieved by employing formal analysis to assist humans and manage risk. This increases trust and reduces validation time, useful in complex systems and where high-confidentiality data may be deployed in low-confidentiality enclaves (or in other high risk scenarios).

We believe rapid recertification can be enhanced by documenting assumptions in tool-supported frameworks where assumptions that continue to hold may be reused. One approach is to use lightweight formal analysis to model and validate security requirements. This analysis should focus recertification efforts on only those requirements that may conflict with IA policy. Furthermore, such conflicts should be resolvable through reconciliation

strategies in which changes to the system can be checked against conformance to the IA policy, without requiring a complete review of the entire system.

In this paper, we present our methodology which principally focuses on modelling and validating specifications of data flow as the basis for evaluating IA policy. The method permits automated analysis of data flowing into and out of a system or component to detect conflicts in the data's specified purpose, which illuminates potential areas of non-conformance and further expedites conflict resolution in regard to the recertification process. We also classify and present reconciliation strategies for resolving the different types of conflicts that may occur. We show that our method is scalable to permit analysis of large, complex, and evolving systems—whose specifications can involve numbers of policies no longer tractable by manual analysis alone.

Organizational granularity refers to the view at which a policy or design artifact is intended to represent its context within an organization. Our scalable, automated analysis can be repeated rapidly as specifications change with the aim to reduce recertification time at any point in the software lifecycle and at any level of organizational granularity. For example, design artifacts detailing collections of software components represent a very high degree of organizational granularity. Networks and departmental interconnections represent a medium degree of organizational granularity. Enterprise-level architectures and inter-organizational connections represent a very low degree of organizational granularity. From the software perspective, these conventions are often referred to by their "level," which comprises the same notion as organizational granularity but on an opposing scale: Software components and their implementations are referred to as low-level artifacts. Network diagrams, detailed design documentation, and inter-departmental processes are referred to as mid-level artifacts. Inter-organizational processes and enterprise architectures are referred to as high-level artifacts. Discussions which must be paired with their context for policy or design artifacts in this paper are referred to by their organizational granularity, for consistency.

Changes to information systems include the addition of new features or new behaviors, or the establishment of new connectors between existing system components and others. Thus, we can envision a range of scenarios that yield characteristically different recertification challenges. For example, designing new systems to replace legacy systems (early lifecycle), integrating new systems with existing systems (mid-lifecycle), reworking existing systems to perform new functions (mid- to late-lifecycle), and during perfective maintenance tasks (late-lifecycle). During these lifecycle stages, software systems are increasing in size due to the nature of these changes, and this increases the time and cost for validation and recertification.

Unfortunately, the naïve approach of dividing up the software system and parallelizing recertification tasks is a separate and more costly challenge that does not gain traction over the problem. Decomposing the system in ways that violate architectural and source code artifact boundaries, such as software interfaces, can increase accidental complexity of certification tasks (Brooks, 1995) when systemic quality attributes, such as ensuring confidentiality, cut across these boundaries. For this reason, our approach employs the notion of a context that may cover a specific application, called an application profile, or an entire secure enclave. Within this context, we can reduce data flow analysis from across enclaves to conform to the same expression and reasoning needed to detect policy conflicts within an enclave. This level of reasoning would correspond to a high or medium level of organizational granularity. In Figure 1, we present two secure enclaves, "A" and "B," with a support service connected to enclave "A" and a handheld application connected to enclave "B." The links that connect these two enclaves are assumed to be

secure as they are within the same operational environment. However, bringing in support services and mobile devices from the outside context into these secure environments represents a change in functionality, which may be a recertification trigger. These recertification triggers are discussed later in our Evaluation section.

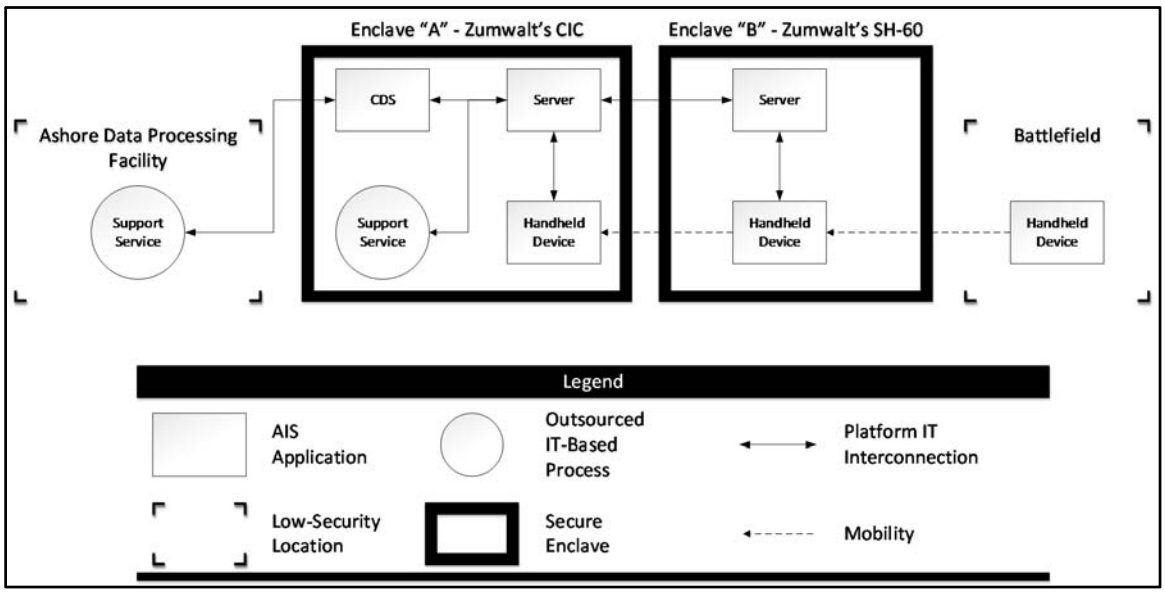



**Figure 1. Example of DoD Distributed IT System Illustrating the System Decomposition Boundary Between Secure Enclave and External Areas**

In this simplified context, we focus narrowly on data actions that access and make available information and postpone addressing questions about which specific security mitigation is needed to address a specific vulnerability. For example, answers to questions about when to use encryption correspond to sensitivity of data and under what contexts that data is made available, which we account for. However, the question about what level of encryption to use is not a central focus of our method, and is well documented in IA policy based on data sensitivity. We now discuss the technical background to our approach.

## Technical Background

The Bell-LaPadula model simplifies the characterization of information flow from low confidentiality to high confidentiality, but not vice versa (Bell, 2005). This model has long been the traditional view of enforcing multi-level security policies in government and military applications. In our method, we characterize the purpose for which information is used as the security level and then allow policy authors to express compositions of security levels through containment and disjointness, for example, a security level may contain or be disjoint with another level. This formalism extends our prior work on semantic parameterization (Breaux, Anton, & Doyle, 2008) for expressing actions on data in Description Logic (DL) as a composition of actors, objects, and purposes, and for transactions involving data, the source and target of the transaction. More recently, we developed a human-readable SQL-like language for expressing these application profiles (Breaux, Hibshi, & Rao, 2014), which we refer to in this paper as the "Application Profile Language." Application Profile Language syntax is parsed and compiled into the Web Ontology Language (OWL), which is suitable for computer processing by an automated DL reasoning tool (Bechhofer et al., 2004).

Subsumption is a syllogism in which one concept describes a more general class of another. This is commonly referred to as a superclass/subclass relationship. In DL, we can check whether one data action subsumes another action, which means every interpretation of the second action is contained in the set of interpretations of the first action. Subsumption allows us to detect conflicts, in particular, when one action is deemed permissible and the same or a subsumed action is deemed impermissible. The relationships between concepts expressed in a specification are mapped directly into the DL model. For example, through subsumption, high-confidentiality purposes for a data transmission may include "top secret" or "for mission-critical purposes," or any number of other concepts which are desired to express with respect to crafting a data requirement.

The DL model is comprised of two parts:

1. an ontology in which key terms are defined, including information type categories and type compositions expressed using subsumption
2. a set of rules governing collection, usage, and transfer of data to third parties

For each data action, the purpose for the action and the party from whom that data is sourced is stated, and for transfer, the party who will receive the data is stated. Actions may be expressed as permitted or prohibited in the application profile. The rules expressing permissions or prohibitions in the profile represent the high-level specification of what actions an application is permitted or restricted from performing, whereas the implementation would entail mapping these actions to functions in code, such as database queries or radar telemetry-based analysis, and so forth.

Figure 2 is a process diagram which illustrates the recertification process steps using our tool. The highlighted area on the right of the figure spotlights the novel contribution of this paper, which is the conflict reconciliation strategies we present later. The recertification challenge that we focus on herein concerns how to compose systems of systems that collect, use, and transfer data across system boundaries, between secure and unsecure enclaves.

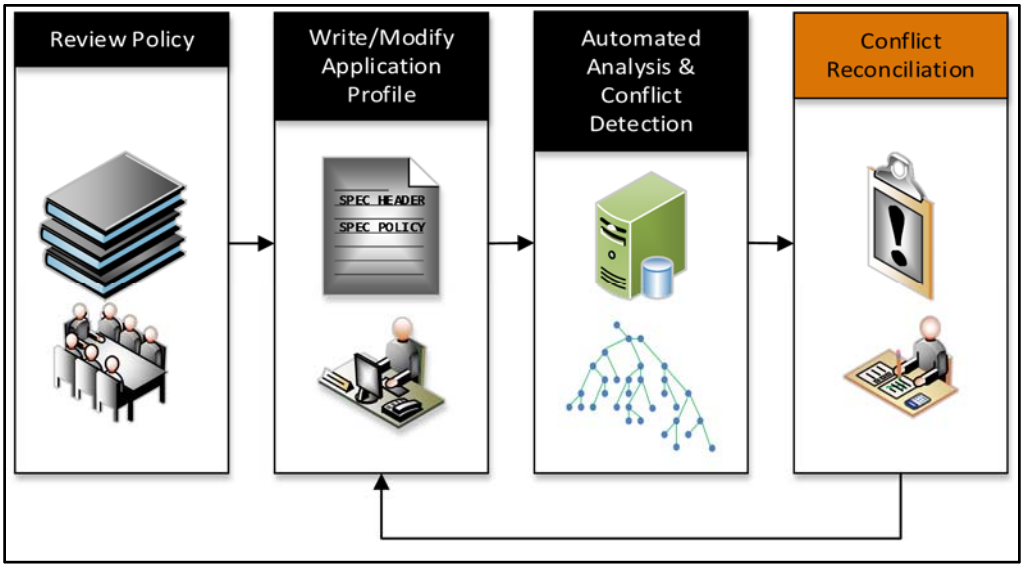



**Figure 2. Rapid Recertification Process Diagram**

## Approach and Running Example

We now present a running example that we use throughout the paper. The example draws from public accounts of real-world vessels and technologies that are currently undergoing sea trials—specifically, we chose scenarios regarding Zumwalt-class destroyers, which we generalize and refer to as "Zumwalt." The running example is used to illustrate the high-level certification concepts that exist in real software systems aboard sophisticated platforms, such as those integrated into Zumwalt's Total Ship Computing Environment Infrastructure. Certification concepts deal with software requirements, and are not reasoned about based on lines of code, given the complexity of software. Zumwalt's electronic systems are comprised of approximately 6 million lines of code—far too much to reason about at the low level, pedantically analyzing each line to determine if it meets requirements or not. Software must be analyzed in terms of high-level concepts, and certification auditors must align analysis methods to meet the same high level of abstraction.

To help achieve alignment between the level of abstraction for our analysis and high-level abstraction in software, details unrelated to software requirements about the ship-class have been removed. Details about the software requirements themselves have been generalized in order to make the focus of our discussion centered on the essential transitive qualities of data. We are not concerned about the underlying functionality or supporting systems of compartments aboard; rather, we are focused on understanding and analyzing systemic behaviors that specify to consume or move data to a certain place, for a certain purpose, to achieve requirements. Rather than being concerned about the memory chips, logic boards, networks, and radiation-emitting or receiving hardware that comprise the air- and surface-borne vessel sensing capabilities aboard a ship, we can abstract this entire sensory capability as the *radar system*. We can reason about the broad concepts of radar data that may be produced by such a system. The application profiles which we use to formalize such concepts give us the capability of automated reasoning at this level. In Application Profile Language, the radar system would be written as an actor. The radar data would be expressed as a data concept. Requirements, expressed through policies inside application profiles, provide the necessary details for determining what is being done with this data in the context of expressed actors. The contents of an application profile can therefore be articulated in notional scenarios based on our running example that captures the necessary details to express an intended new system functionality.

Our running example is designed to be illustrative of high-level software concepts, and may not reflect the entirety of the underlying details of a vessel, the bits and bytes of data in their onboard computer systems, or a platform's true capabilities. However, the example provides sufficient detail about where and what data is being used, so that we can reason about data's transitive properties given the general characteristics of these systems. Without worrying about the lines of code that comprise these software systems, we can reason about how data is used, and how this data may need to be transferred outside of a vessel's enclave to another party to fulfil requirements.

In our running example, Zumwalt's key capabilities that we have abstracted correspond to some of the platform's distinguishing features compared to other destroyers; these include Zumwalt's advanced radar system (Tolley & Ball, 2014), abstracted as the *radar system*. Zumwalt's sensor-netting capabilities that permit sharing of information with friendly platforms (O'Neil, 2007), including AEGIS technologies or the Total Ship Computing Environment Infrastructure, are abstracted as the *information sharing mechanism*. Together, these two abstracted systems permit enough expressivity in terms of the data concepts that exist in the real world so as to maintain realism with respect to DoD IA policies and software certification processes. We reason about requirements for these computerized systems

using already established directives for security used in practice, such as DoD Directive 8500.01. Zumwalt may be considered to be in any theatre, with no specific mission, which permits our example to be illustratively modified and re-examined in different operational contexts. In any of these contexts, the underlying thought processes that are required to analyze decisions about adding new features, or modifying connections with outside information systems and processes, are the same. This focus on only information sharing concepts rather than technical implementation details also means that these scenarios could be extended to cover any vessel, or land-based system.

The Zumwalt-class' software implementations have made extensive use of real-time operating systems with stratified and segmented "high" and "low" security networks as part of the Total Ship Computing Environment Infrastructure (Lynxworks, 2007). These technologies conflate directly with the Bell-LaPadula (Bell, 2005; Landwehr, 1981) view of IA and security, and is the same model for purpose which we show in our DL ontology. Attention on software aspects of the class' radar system and information sharing capabilities is an identified concern in design reviews of the vessel (O'Rourke, 2012), emphasizing the need for reducing software certification costs and time, which contribute to the expenses associated with information sharing capabilities such as sensor netting. Certification becomes an even larger problem when considering information integration with multiple platforms or allied vessels in the demanding context of future battle spaces, which is the intent behind our selection of Zumwalt as the subject of our example.

## Profiles, Conflicts, and Tracing

We now describe our approach to trace data flows in and through application profiles, and to detect conflicting requirements. When policies covering multiple organizational granularities are used in a single application profile's policy, overlapping policies may result. In high and medium granularity overlaps, organizational policies may conflict with individual policies governing software components, and vice-versa, for example. In low and medium granularity overlaps, inter-organizational policies may conflict with departmental policies. High and low granularity overlaps are also possible, in which inter-organizational policies may conflict with those governing software components. In general, overlapping policies may lead to conflict, and these conflicts must be reconciled in order to prevent data spills. Later in this paper, we discuss identifying and reconciling specific types of conflicts that exist as a result of these overlaps in light of this risk.

### *Actions on Data*

In our model, application profile rules may govern three primary actions which express the transitive nature of data as it moves through a system. These actions are collection, usage, and transfer of data. Each action concept has assigned roles that relate the action to actors, data, and an associated purpose. The collection action describes an act by a party to access, collect, obtain, receive, or acquire data from another party. The usage action describes an act by a party to use or consume data in any way. Transferring describes an act by a party to transmit, move, send, or relocate data to another party so that they may collect it. Pairs of rules that permit collections and transfers on the same datum are referred to as a *data flow*. When one of the rules in the pair is expressed with respect to a third party, this data flow can be *traced* to the third party.

For example, Zumwalt may need to transfer information about the presence of an enemy radar contact to a friendly vessel in the vicinity in order to initiate a combined engagement. In another scenario, a friendly vessel may need to transfer the same information to Zumwalt so that Zumwalt can engage it. In both scenarios, these actions performed on the radar contact data represent collection, transfer, and usage actions.

Collection actions generate radar data from both vessel's radar systems, and permit one vessel to receive data from another. Transfer actions permit one vessel to share its data with another. Usage actions permit a vessel to engage the radar contact once the data is within its enclave.

Actions are further described by DL roles. The *hasDatum* role represents the action's affected datum. The *hasActor* role indicates the source actor from which the data was collected. The *hasPurpose* role indicates the purpose for which an action is being performed. These purposes, in line with the original Bell-LaPadula model, are abstracted as *high-confidentiality* and *low-confidentiality*. These classes of purpose may be further subsumed by any other purpose in order to extend the ontology to fit the policy expressivity needs of a specific organizational granularity.

### *Finding Conflicts in Policies*

We now describe how the rules governing actions on data may conflict, and how we can detect these conflicts automatically. In our DL ontology, a conflict is defined as an instance when an action is permitted by one rule, and prohibited by another. Rules governing the permission and prohibition of specified actions are described using the Application Profile Language, which is parsed by our language parser into Web Ontology Language (OWL). OWL can be analyzed automatically by freely available open source DL inference engines such as Pellet, Fact++, Racer, and HermiT. These DL reasoning engines have historical pedigrees in academic usage for formal analysis (Baader, Horrocks, & Sattler, 2005).

In the DL representation of an application profile, a rule must be determined by the reasoning engine as equivalent to both a *right* (also known as a permission) and a *prohibition*, in order to be found equivalent to a *conflict*. Both the right and the prohibition must act over the same datum and for the same purpose in order to be equivalent.

An equivalence relationship requires equivalence in both directions, as one might find in a mathematical expression (such as $a + b = c$). In some cases, the conflict can only be reasoned about with respect to one rule, because reasoning can only occur on "one side" of the equivalence operator. This is explained through subsumption. We cannot distinguish the rule on the other side of the equivalence as truly involved in a conflict; there may be further subsumed interpretations using subclasses which do not conflict.

Using our running example, we may imagine an application profile which governs how Zumwalt is permitted to share radar data that it has collected. In this case, radar data contains data about enemies, friendlies, and terrain. In DL, radar data subsumes data about enemy vessels, data about terrain, and data about friendly vessels. When radar data subsumes these more general concepts of data, radar data is the subclass, and data about enemies, terrain, and friendly vessels are the superclasses. The DL reasoning engine must reason about all of the superclasses of a datum when reasoning about the subclass, and different rules may exist that govern each of these data.

We instantiate this example with a profile that contains data definitions and a basic policy for Zumwalt's radar and information sharing, seen in Figure 3, and conforming to the profile language syntax defined in Breaux et al. (2014). This figure also shows the abstraction of Zumwalt's radar system and information sharing system as actors. On the top of the figure are the data definitions that correspond to the header section of the profile. On the bottom of the figure is the policy, which is comprised of five rules. Together, the header and the profile of a policy comprise a full application profile. Each data definition and rule seen in Figure 3 is annotated with the English language representation of the Application Profile Language syntax.

This profile has been seeded with a conflict that is derived from the intended system functionality that would permit Zumwalt to share collected radar data with friendly fleets for low confidentiality purposes. The intention of this policy is to provide information sharing on this vessel, but this conflicts with an overarching organizational policy that does not allow data about friendly vessels to be shared with other friendly vessels. We now describe the nature of this conflict and how it is detected by our approach.

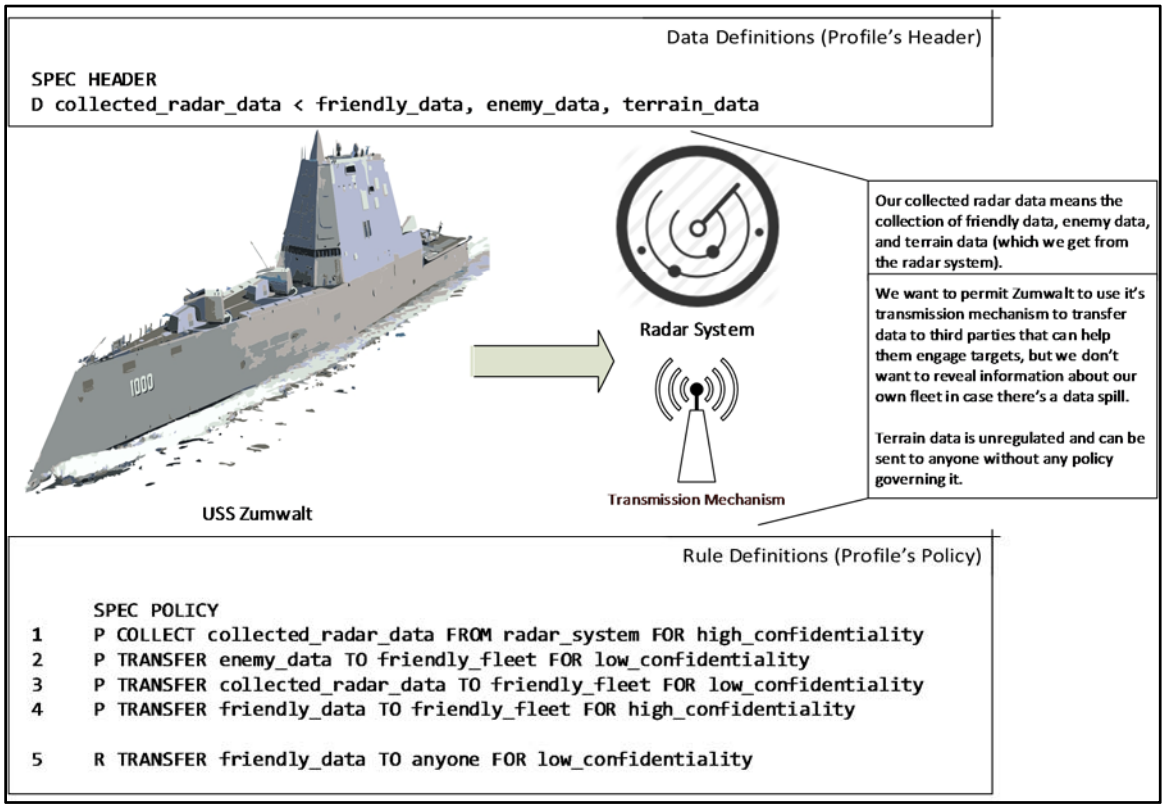



**Figure 3. Representing Zumwalt as Two Abstracted System Components (Actors), With an Associated Application Profile That Describes the Rules That Govern the Data Produced and Transferred by These Actors**

If we write a policy that permits us to share all radar data, but prohibits sharing data about friendly vessels, we cannot share all radar data. This is a one-sided conflict. This conflict occurs because the relationship between these general data concepts and the specific radar data subclass is actually defining a composition rather than a subsumption; however, the semantics of the relationship are the same when expressed in this way. As a result, we may not be able to reason about the superclass of general data that we have defined here, but we can reason about the subclass of radar data, and its relationship with friendly data. Superclasses and subclasses are not equivalent, but we can infer enough about radar data to know that the policy would still generate a conflict—this reasoning about the subclass rather than the superclass is similar to reasoning about one side of an equation based on inferences about the other side (hence the name *one-sided* conflict).

One-sided conflicts are highlighted by the DL reasoning engine on one rule. This is convenient for our purposes because this single rule is the one that must be acted upon using the conflict reconciliation methods seen in the next section. The other conflicting rule, governing the superclass, cannot be reconciled using the approaches we have identified

due to the nature of this one-sided conflict. We cannot say for certain whether we should perform the reconciliation action on the “other side” of the conflict since we cannot say that it conflicts in all possible subsumed interpretations. An example of such a conflict is seen in our running example.

We infer that collection/transfer data flow traces are functionally the same as collection/usage traces, in terms of the transitive properties of data. However, given that the scope of our analysis is to determine where data spills may occur due to information transmission to third parties, collection/usage data flow traces are not relevant. Collection/usage flows can only occur within the system bounds of a single party. A party may consist of more than one secure enclave. Collection/use traces may still generate conflicts, but these conflicts would indicate mismatch of an intended purpose within a secure enclave, and do not impose a serious risk of data spills unless there is a separate rule permitting the transfer of this datum elsewhere.

## Conflict Reconciliation

When conflicts arise, we have identified two main strategies which work to match the generality of purpose. Matching the level in the class hierarchy of subsumed purposes reconciles conflicts in which a more specific high-confidentiality purpose is permitted, but more general low-confidentiality purposes are prohibited. This serves to mitigate the likelihood that data will be transferred outside of the secure enclave without explicit authorization for a specific high-confidentiality purpose. Matching purposes also serves to mitigate being mismatched with a third party that will consume (or retransfer) this data for a more general purpose, which would constitute a data spill. The reconciliation actions which we have identified, *redaction* and *generalization*, serve the same purpose as their namesakes in legacy document-oriented processes. Both actions serve to transform data in such a way that it is permissible to transfer it for low-confidentiality—and therefore more general—purposes.

### *Redaction*

Redaction means to remove elements from a collection of data in order to limit the spread of information that must remain only within the secure enclave for some specified purpose. In effect, our conceptualization of redaction in data flows within our process is the same. In the context of reconciling conflicts across data flow policies, redaction can be performed on any datum which is itself a collection of subsumed data. In our ontology, this means the collection is a superclass. This is because the act of redacting data eliminates one or more subsumption relationships between the collection datum itself, and the subsumed data types. Redaction results in a new, redacted datum which is fit for a more general low-confidentiality purpose, as compared to the original datum which was only suitable for a specific high-confidentiality purpose.

As per our running example, Figure 4 shows a basic profile that has been retrofitted to permit Zumwalt to share radar data with a third party, which is a friendly fleet. The definition for *radar data* in this instance refers to data about friendly vessels, data about enemy vessels, and data about terrain/surface objects, which is expressed through the subsumption relationship seen on the left side of Figure 4. Internally, Zumwalt collects this radar data within its own secure enclave from the vessel’s onboard radar system, and this data is intended for consumption or transfer with an unspecified high confidentiality purpose only. It should be noted that in this scenario, the policies which govern consumption of this data internal to the Zumwalt are not within the scope of this discussion, as they are not related to the risk of data spill during transfer to a third party, and have been left out of the profile for simplicity. The area of security interest for our analysis in this scenario is with

respect to the requirements that govern how this data may be transferred to third parties outside of Zumwalt's secure enclave, where the risk of data spill does exist.

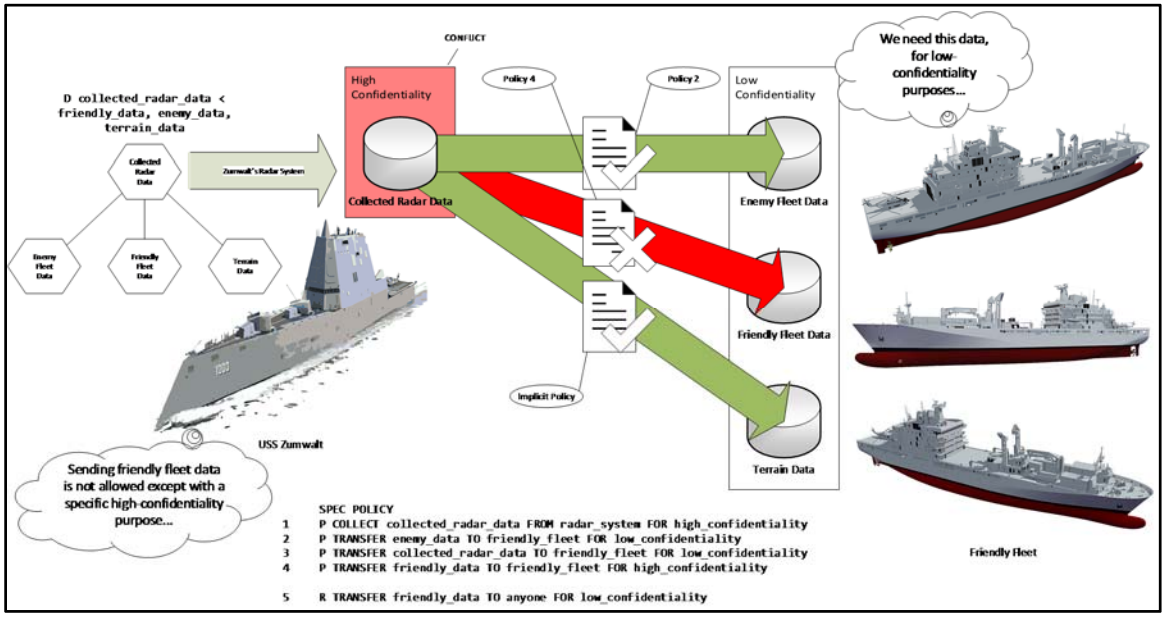



**Figure 4. Simple Radar Data Sharing Profile for Zumwalt**

Next, we break down the individual rules that are seen in the profile introduced in Figure 3. The conflict is highlighted in Figure 4. Here we unravel the conflict and show how the redaction mechanism can be applied. First, we show the English language interpretation of a rule. Below, we show the application profile language used to express this rule, and the corresponding formalization in DL which is generated by our parser.

1. Permit collection of collected radar data from Zumwalt's radar system, designating it as high-confidentiality data.

| Application Profile Language | Formalization in Description Logic |
|---|---|
| P COLLECT collected_radar_data FROM radar_system FOR high_confidentiality | $T \models p0 \equiv COLLECT \sqcap \exists hasObject.$ collected_radar_data $\sqcap$ $\exists hasSource.$ radar_system $\sqcap$ $\exists hasPurpose.$ high_confidentiality |

2. Permit transfer of data about enemy vessels to friendly fleet members for general, low-confidentiality purposes.

| Application Profile Language | Formalization in Description Logic |
|---|---|
| P TRANSFER enemy_data TO friendly_fleet FOR low_confidentiality | $T \models p1 \equiv$ TRANSFER $\sqcap \exists hasObject.$ enemy_data $\sqcap$ $\exists hasTarget.$ radar_system $\sqcap$ $\exists hasPurpose.$ low_confidentiality |

3. Permit transfer of all collected radar data to friendly fleet members for general, low confidentiality purposes. *This rule generates a conflict, which is explained below.*

| Application Profile Language | Formalization in Description Logic |
|---|---|
| P TRANSFER collected_radar_data TO friendly_fleet FOR low_confidentiality | $T \models p2 \equiv$ TRANSFER $\sqcap \exists has Object.$ collected_radar_data $\sqcap$ $\exists has$Target. friendly_fleet $\sqcap$ $\exists hasPurpose.$ low_confidentiality |

4. Permit transfer of data about friendly vessels to friendly fleet members for specific, high-confidentiality purposes.

| Application Profile Language | Formalization in Description Logic |
|---|---|
| P TRANSFER friendly_data TO friendly_fleet FOR high_confidentiality | $T \models p3 \equiv$ TRANSFER $\sqcap \exists has Object.$ friendly_data $\sqcap$ $\exists has$Target. friendly_fleet $\sqcap$ $\exists hasPurpose.$high_confidentiality |

5. Prohibit transfer of friendly fleet data to anyone for general, low confidentiality purposes. *This rule conflicts with Rule 3, explained below.*

| Application Profile Language | Formalization in Description Logic |
|---|---|
| R TRANSFER friendly_data TO anyone FOR low_confidentiality | $T \models r0 \equiv$ TRANSFER $\sqcap \exists has Object.$ collected_radar_data $\sqcap$ $\exists has$Target. Actor $\sqcap$ $\exists hasPurpose.$ low_confidentiality |

Rule 5 works to prevent information about the friendly fleet leaking to third parties. This implies that regardless of the target for the data flow, a high-confidentiality purpose must be specified to justify friendly fleet data leaving the secure enclave of the Zumwalt. This requirement is instantiating a normal compartmentalization strategy for managing the flow of information, but this rule is in conflict with the intended new functionality. The retrofitted functionality requires the collected radar data to be shared with friendly fleet members, since collected radar data has been defined as friendly fleet data, as well as enemy fleet data and terrain data. This conflict can be resolved through redaction, as the prohibition only restricts the transfer of friendly fleet data for general, low-confidentiality purposes. By redacting this datum from its relationship with the collected radar data concept (given that it is the superclass for the other three types of data specified), we can define a new datum. Redacted radar data only contains enemy fleet data and terrain data, which are both unrestricted by purpose to high-confidentiality. The redacted datum may instead be used for low-confidentiality purposes as permitted by Rule 2), or implicitly permitted due to there being no rule which exists in this profile's policy that governs the flow of terrain data.

Thus, the redaction statement syntax appears as follows:

| Application Profile Language |
|---|
| P REDACT(collected_radar_data -> redacted_radar_data, friendly_data, low_confidentiality) |

The above syntax represents the original datum to be redacted (linked with the -> operator to the new datum definition), the concept in the subsumption relationship to be removed from the original datum, and the newly established purpose for the redacted datum. This then permits the modified Rule 3 to read as follows, after resolving the conflict:

3. Permit transfer of all redacted radar data to friendly fleet members for general, low confidentiality purposes.

| Application Profile Language | Formalization in Description Logic |
|---|---|
| P TRANSFER redacted_radar_data TO friendly_fleet FOR low_confidentiality | $T \models p2 \equiv$ TRANSFER $\sqcap \exists has Object.$ redacted_radar_data $\sqcap$ $\exists has$Target. friendly_fleet $\sqcap$ $\exists hasPurpose.$ low_confidentiality |

There is an alternative conflict resolution approach which is functionally the same as redaction, and may be more or less desirable depending on the functional context of the requirements. If there had been a high-confidentiality purpose specified for sharing the friendly fleet data in Rule 3, then the conflict would not have existed since Rule 5 only restricts the transfer of this datum for general low-confidentiality purposes. By narrowing the allowable purposes for which the data can be transferred and therefore consumed by a third party, the policy expressed in the profile would have been in alignment with the new requirements to transfer the radar data. However, there is a strong likelihood that the narrower range of purposes permissible for this datum may not be general enough for the third party's intended purpose, especially if they had expressed a broader (low-confidentiality) intended purpose for the subsumed data.

A simple scenario which illustrates this case may be that the friendly fleet's policy had intended, by their system's internal design, to communicate the location of the enemy fleet to nearby civilians somehow. This action would still violate Rule 3 with respect to Zumwalt's requirements since the enemy fleet data also includes collected radar data, and is therefore subject to being restricted to high-confidentiality purposes only, as low-confidentiality purposes are prohibited for transferring data. Due to the policy expression having deliberately chosen the more inclusive datum concept of all collected radar data, the previous resolution is to redact the datum and remove the general relationship with friendly fleet data.

The functionally similar alternative reconciliation is to completely recreate new policies which govern the data differently and express prohibitions and permissions uniquely for each individual datum concept. This secondary approach, while functionally the same as the action of redacting the original datum, may result in lengthy policies which do not gain expressiveness over the simpler act of redaction, and does not require completely redefining the data entities that are expressed in the original profile.

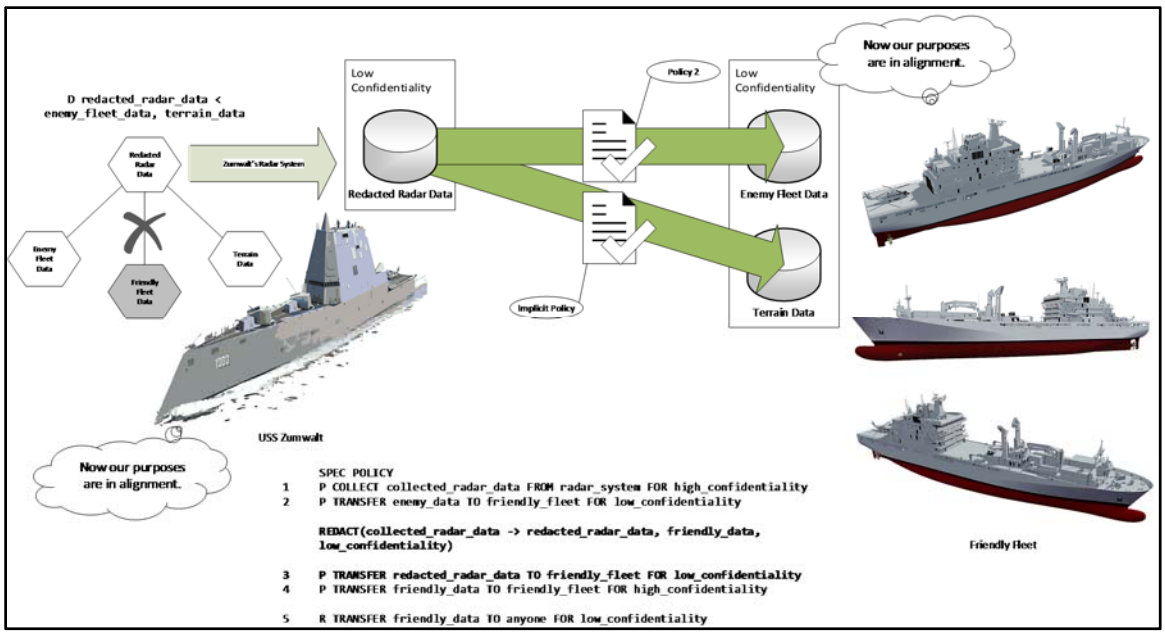



**Figure 5. Transmission of Redacted Datum**

### *Generalization*

Generalization refers to the process of capturing a broader notion of a concept using inferences from a series of specific cases. In this context, generalization refers to the process by which a datum is transformed to represent a far broader notion of the original information, retaining enough precision to be useful, but not supplying enough precision to infer the original data. Examples in common usage include generalizing a position into a region, rather than a series of coordinates (such as Pacific Ocean, which is far more general, versus 0.0000° S, 160.0000° W, which is far less general) in order to obscure the precise location of a fleet by generalizing it to include a much broader area. The level of specified measurement precision selected for generalization must be appropriate to the datum. The act of performing generalization differs from redaction since the original ungeneralized datum cannot be a collection, or the generalization must be performed on all members in the collection. Generalization may also not have the same subsumed relationships as those applicable to the data undergoing redaction.

Generalization uses the same conflict resolution mechanism as redaction, in that it acts to realign the specified purposes for which a datum may be collected and transferred (or collected and used). For example, friendly fleet data such as locations may be generalized to express only regions rather than precise coordinates, in order to permit usage under low-confidentiality purposes among the collection of other data, as seen in Figure 4. In doing so, the superclass of what constitutes collected radar data is modified, and all subclasses in the collection which have been expressed already in the profile become permissible for use with low-confidentiality purposes. This is because friendly fleet data, enemy fleet data, and terrain data have all been generalized under this reconciliation strategy; each datum that is part of the superclass has been transformed and redefined as a new datum. These new data are implicitly permitted for transfer under general low-confidentiality purposes, just as terrain data had already been permitted without modification, as seen in Figure 4 and Figure 5.

### *Merging: Undoing Redaction and Generalization*

Aside from simple differences in interpretation of policies and the conflicts that may result, our findings show that designers may accidentally introduce conflicts through actions that transfer data with mismatching purposes to those of their third party collector (or vice-versa) in order to realize system requirements that do not obviously conflict. This may occur in instances where the original design of a system did not consider collections of data to be usable as separate data when removed from the collection, as in our running example with collected radar data. In this example, the nature of the separate data in the collection was obscured by the subsumption relationship. The purpose of data which is combined from separate data can also be obscured. Merging disjoint data for more general purposes can create a new single datum that should only be used for the same generality of purpose as the original separate data. If this data is used for a more general purpose than any of those originally specified for any of the recombinant parts, there is a risk of repurposing data in a way that violates the original intention of the policy, which can lead to data spills. In our methodology, we define this act as *merging*, which requires two or more data to be used in conjunction with one another to create a new datum.

Given the transitive nature of data, and the inability to reason about data which moves past the designed system boundary of the protected enclave, proactive security assumptions must be made such that possible recombination of data may occur in any instance in which data has been transferred outside of the enclave for a general low-confidentiality purpose. This transfer and its associated risk of recombination with other data holds that the data may possibly be used for any similarly low-confidentiality purpose in conjunction with all other data of the same level of purpose. Conversely, if a datum is transferred for a specific high-confidentiality purpose, it may not be used for a low-confidentiality purpose without violating a policy or generating a conflicting requirement that must be reconciled.

The merge act combines one or more related data to create a new datum. This new datum may be equivalent to a datum originally specified for a high-confidentiality purpose, but had been previously redacted. The outcome of a merge act may yield more information than the original high-confidentiality purpose of a constituent data point. For example, a datum describing the position of a destroyer fleet at a time *T* only has the power to assert that the fleet was at that position at that time, which may be sufficient for some specific low-confidentiality purpose, such as establishing a point of rendezvous during a fleet maneuver. However, possessing additional data about the position of a fleet at time $T + 1$, $T + 2$ and so on can yield sufficient information to determine the previous bearing of the fleet and/or the likely future course of the fleet. This additional information may be restricted to a higher confidentiality purpose than was unintended for the data by its original specification, which could serve to mitigate the likelihood of merging this data without a specified high-confidentiality purpose. Generalization of this data may also serve as a countermeasure for this merging, since less precise coordinate data at all of these times would render an adversary unable to track the fleet precisely enough for the data to be useful. Increasing the amount of data points that are merged together in this basic example can further increase the precision or predictive power for a determination of the fleet's movements, even in the presence of course corrections or evasive maneuvers which attempt to obscure the true intent of the fleet. This may also counteract the generalization strategy if some data points have been generalized while others have not, because the data flows transferring generalized data versus the ungeneralized data had low- and high-confidentiality purposes specified, respectively.

The same reasoning applies to other compositions of data with different information, even including the data that are not time sensitive. For example, determining the individual composition and outfitting of an individual ship in a fleet may yield only some tactical information about how Zumwalt's executive officer may assay it as a single threat in isolation. However, finding similar data about all ships in a fleet and combining this information into a newly merged report about the whole fleet configuration may yield important clues about the strategic significance and intent of the fleet, far beyond the original tactical information of any one given ship's threat.

In order to shield against an adversary performing this same merging of disjoint information to their advantage, the DoD must safeguard against the likelihood that data flows with specific purposes are repurposed for more general, low-confidentiality purposes. Otherwise, there is a risk that similarly purposed data flows may be merged.

Merge-related conflicts may be detected by determining the generality of the purpose in which data is transferred outside of the secure enclave and to a third party, which is a typical IA strategy. One general principle of IA strategy is that information should be disseminated on a need-to-know basis (based on the purpose of its usage, as per Bell-LaPadula), and those who do not need to know will not receive the data in question. Tracing data flows and matching the generality of the specified purpose is key to determine if a datum may be used for a different purpose elsewhere. Recall the example seen in Figure 5: If the friendly fleet had seen fit to recompose the original, non-redacted radar data using the redacted radar data in conjunction with the terrain data and enemy fleet data that they received, Zumwalt's data sharing policy would be in violation due to the friendly fleet's merge action.

Sending data for a specific, high-confidentiality purpose means that the collection actions that correspond to a datum's transfer from the perspective of a third party must also match our high-confidentiality purpose, as must all subsumed classes of data related to the datum in question. If there is a mismatch in purpose, then there is a risk that any of the subsumed data may be repurposed for a more general purpose, which then increases the risk that it will be recombined with other subsumed data for the same superclass. This mismatch is in itself a conflict, as it violates the specified purposes expressed in the profile, and can be identified in the model by using our conflict detection methodology. This pinpoints the specific policies that are in conflict, and these policies become the subject of our conflict reconciliation strategies.

## Evaluation and Identifying Recertification Triggers

Based on our running example, we surmise that conflicts may arise as a result of unintended or implied actions in a profile designer's expression of data as a collection, which we see as a direct result of the new requirements imposed by new system functionality. Designers may express policies without having considered the individual elements in collections and the implications of adding new features that require usage at a different level of purpose than the original specification. In our running example, the original definition of collected radar data and the prohibition of sharing any data about friendly vessels implicitly did not consider that some future requirement would need this data to be shared with a third party. This conflict arises as a result of developing profiles that meet current system IA requirements, as well as functional requirements, but these profiles may not be useful as functional requirements change. Over time, operational requirements change, system designs change, and each of these changes may be unpredictable but still necessitates re-evaluation. Therefore, we must evaluate the effectiveness of our approach in terms of conflicts that can be identified and reconciled based on these changes.

We have identified that conflicts may result from overlapping organizational granularities expressed on one profile, and these conflicts must be reconciled. We have also identified that conflicts may result from changes in the intended functionality for a system governed by its profile's policy. Recertification is then triggered by the reconciliation of these conflicts because they necessarily result in further changes to the profile beyond those originally introduced by new functionality. We assert that the recertification triggers are those actions that change a system's profile sufficiently to merit reanalysis of the policies expressed. These actions include (but are not limited to) the addition of new features, modification of an existing behavior or feature requiring new connectors to outside parties, and modifying an existing connection to an outside party to serve a new purpose.

Our running example suggests that adding a new feature to a system is likely the most obvious recertification trigger, as it is functionally the same as modifying existing behaviors, and may also involve new connectors to outside systems for which data flows must be analyzed. In the case of Zumwalt introducing new functionality for its radar and information sharing systems, we could see that conflicts arose because of the mismatch in intended purposes for the previous policy which prohibited sharing of data about friendly vessels. This conflicted with the intended new functionality to share radar data with a friendly fleet, because the original policy was not intended for this functionality given the collection of friendly, enemy, and terrain data within collected radar data. Under circumstances such as these, the conflicts can be reconciled using the strategies we have defined, and profiles can be rapidly re-evaluated using our tool. Any new conflicts that arise may be reconciled using the same strategies until all of the conflicts are eliminated. Under this approach, potential conflicts that lead to data spill risks can be reasoned about prior to the development of the components which act as recertification triggers, or may be remediated if they are found after they have been instantiated as part of a new design or upgrade.

### *Scalability Evaluation*

We must also evaluate our approach in terms of its efficient application and scalability to increasingly large profiles. In order to determine whether this approach is computationally scalable for extremely large compositions of systems, a performance simulation was conducted. This simulation was used to determine how much time was required for the language and DL reasoning engine to reason over profiles and detect conflicts, based on the size of the profiles. The simulation was designed to examine a much larger number of requirement statements than the simplified 5-rule profiles used in our running example, and strived to simulate profiles that would be as large as existing systems' requirements. Previous case studies showed that the number of rules in a policy that we would expect to see in an integrated service scenario in commercial civilian applications is approximately 144 statements, so this provided a basis for determining the extent to which our simulation should scale, but no relative measure for performance.

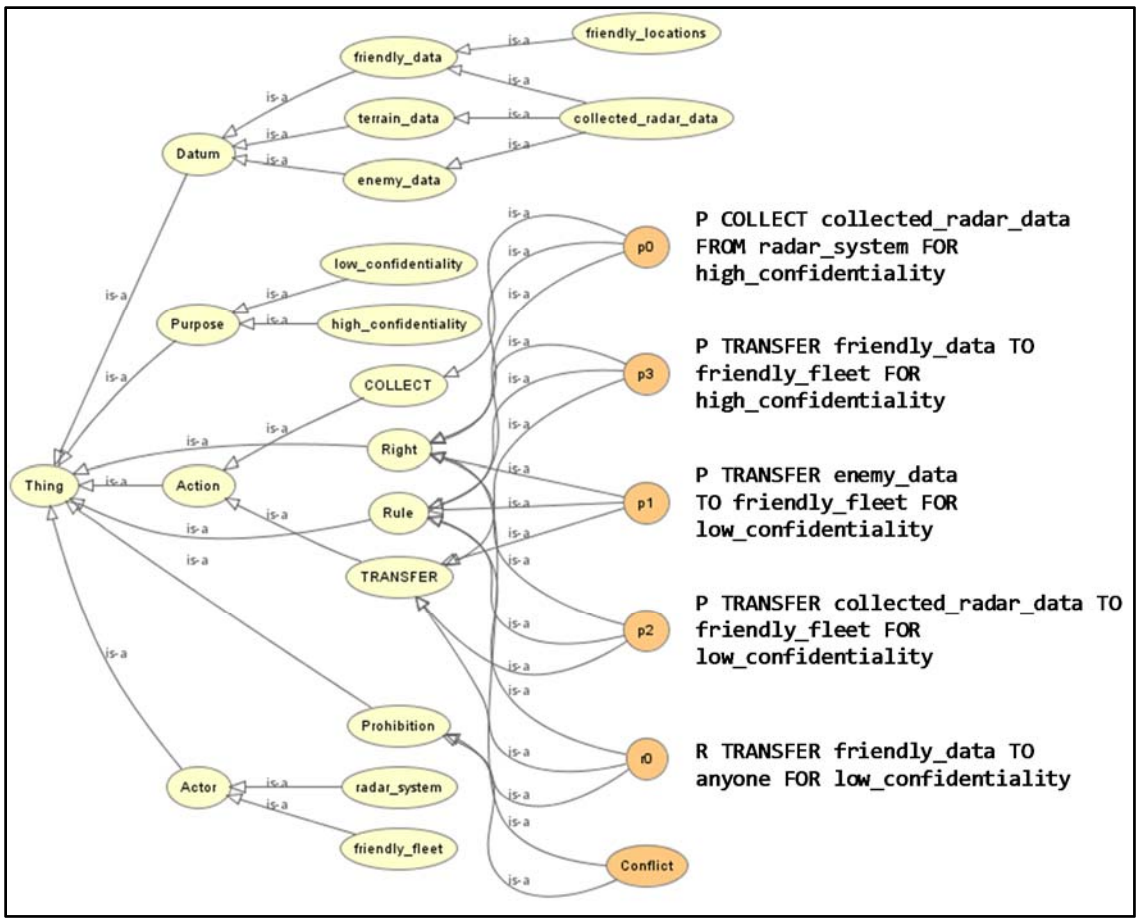
We held the number of concepts and individuals constant and varied the number of rules to determine the time required to reason over the entire profile as the size of the profile increased. The scalability of our approach is largely dependent on the DL inference engine used to analyze the OWL ontology, rather than the tools which parse the Application Profile Language syntax. For the purposes of our study, we used the HermiT reasoning engine, as previous studies had shown it to have the fastest performance (Breaux et al., 2014) compared to other contemporaries (such as Pellet, Fact++, and Racer).

Our simulation results suggest that the parsing process, reasoning process, and output occurs expediently enough to claim that it scales quasi-linearly to analysis of profiles involving hundreds of data flows, and hundreds of rules. Our simulations show that even the

largest profiles within our test dataset could be analyzed in under 400 seconds. We expect that this would be sufficiently little time to permit modification and reanalysis of similar profiles several times in rapid succession. However, with no point of comparison with other tools that perform this functionality, we have no objective basis for which to determine the impact of our performance claims. We can only conjecture that for most purposes, it would be sufficient to be able to reanalyze a real profile of corresponding size to our largest simulated profiles three times within one hour, and use this as a baseline for our performance analysis. This conjecture is based on the fact that that there is substantial cognitive load on the analyst to determine the correct conflict reconciliation approach for the given conflicts they encounter. However, these claims may be substantiated better in controlled experiments that quantify the amount of time required to instantiate each conflict resolution strategy, in order to validate this conjecture and substantiate the overall impact of our performance claim. An exploration of similar methodologies may also yield a point of comparison for the relative speed of our approach.

Simulations were performed in groups of 27 repetitions, in which an ontology was randomly generated as a result of a syntactically accurate minimal profile that expresses generic data definitions and purposes. This profile would have no semantic meaning to an analyst since it does not express concepts within the true problem space of IA requirements, but it is valuable for performance analysis and simulations of the reasoning process since it is structurally similar to real profiles and can be easily varied in size. For profiles of this kind, there appears to be a proportionally increasing probability of a conflict arising to the number of requirement statements and individuals specified in each profile. We found that there is a relationship of approximately 1.13 conflicts found for each increment in the number of rules expressed in the simulated profile for profiles containing greater than 15 rules. This relationship is visible in Figure 8, which is a scatterplot that shows that the number of conflicts increases quasi-linearly with respect to the size of the profile. There does not appear to be a direct correlation between the reasoning times required and the number of conflicts found within the reasoning process; using Pearson's correlation, there is not a statistically significant relationship found with $\{\underline{r}(874) = .36, \underline{p} > .05\}$. Sixteen data types were used uniformly across all simulations, and simulation groups ran with increasing numbers of statements (an increase of two statements for odd numbered runs, and three statements for even numbered runs) from three statements up to 80 statements. The number of actors specified was random and increased proportionally to the number of statements, beginning from two with a maximum of 113, with mean of 59 and standard deviation of 24.

Figure 7 is a scatterplot graphing the increasing size of profiles versus the time required to reason over them. The figure shows that there is a proportional relationship between the time required to reason over the profile and the profile size (which is also the size of the OWL ontology), but aside from some outliers, the largest proportion of reasoning time required remained below 400 seconds even as the number of security requirements increased. Minor outliers appear to be a result of increased conflicts, and extreme outliers may be explained by aberrations in the time measurements resulting from changes in the proportion of processor time allocated to our test environment versus background processes on the same system, rather than the structure of the ontology or the speed of the tool itself. Parsing an entire profile requires less than one second on average. The major portion of the end-to-end processing time is devoted to HermiT's automated reasoning and final output of the detected conflicts.




*Note*. Arrows represent the subsumption relationships between concepts seen in the ontology. This ontology corresponds to the policy seen in Figure 4's profile. Mechanically reasoning over this ontology using a DL reasoning engine permits us to determine where permissions and prohibitions acting on the same (or subsumed) data conflict.

**Figure 6. A Simple DL Ontology, Showing 5 Rules, Consisting of 4 Permissions (p), One Prohibition (r), and 3 Datum Definitions**

In large profiles, there may be hundreds of conflicting requirements. In the presence of hundreds of rules, any collection/usage, or collection/transfer rule pair may increase the likelihood of a conflict. Each conflict must be identified and reconciled in order to mitigate the risk of a data spill. Without automation to find and analyze these conflicts automatically, there is a far more substantial risk that they will be overlooked by humans due to the effort and repetition required to analyze these policies manually. Manual analysis is a long and tedious process that quickly becomes intractable for humans even with small numbers of policies analyzed in isolation. This process necessitates mechanical analysis and automation, only possible through formal analysis using the supporting software tools. An example visualization of the complexity of profile ontologies can be seen in Figure 6, which visualizes five rules and three datum definitions, using the Protégé ontology visualization tool from Stanford University and the University of Manchester. This visualization is the same profile seen in our running example. In order to manually analyze this ontology without tool support, analysts would need to cross-reference each prohibition rule with each permission rule to determine if they acted on the same datum. Then, this subset would need to be examined to see if the prohibitions restricted an action that existed in one of the

permissions. That same analysis would need to be repeated for each subsumed concept that existed for each datum, and for each purpose.

**Table 1. Scalability Analysis Summary**

| | |
|---|---|
| **Average Profile Parsing Time** | <1 second |
| **Largest Profile Size** | 80 rules |
| **Longest Profile Processing Time** | 400 seconds |
| **Average Conflicts per Statement** | 1.13 |

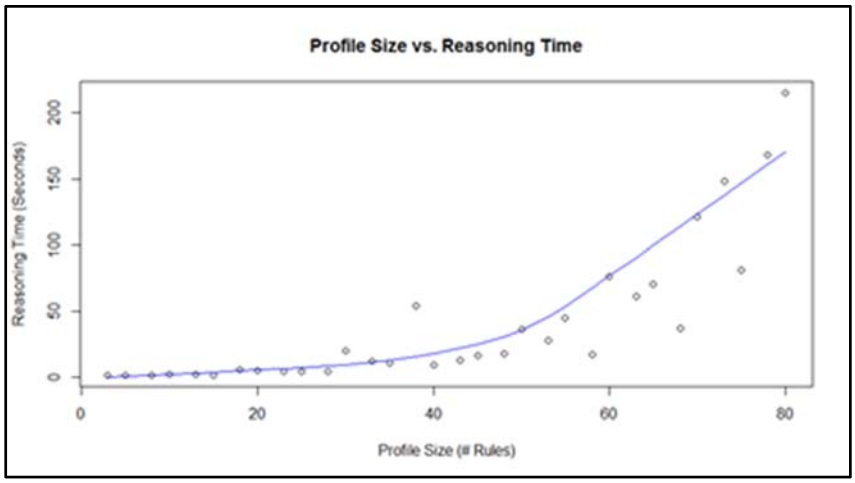
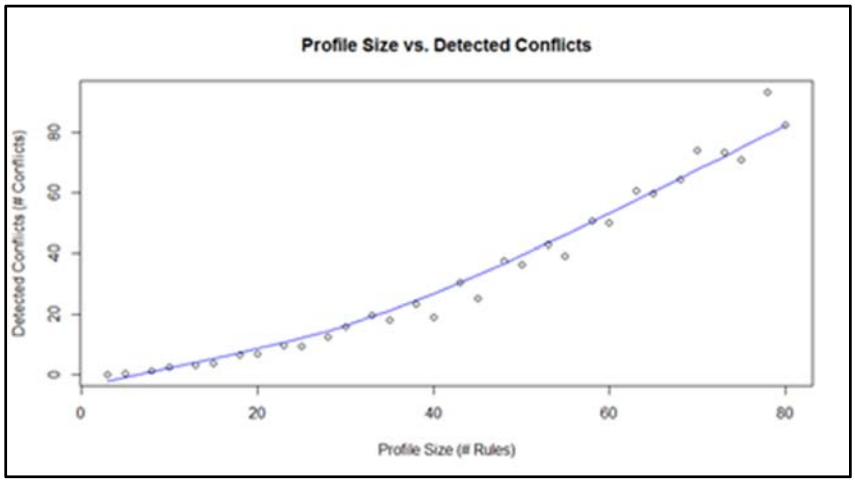



**Figure 7. Plotting the Number of Security Requirements in the Profile vs. the Amount of Time (in Seconds) Required to Reason Over the Entire Profile (and Its Resultant DL Ontology)**




*Note*. There is a clear proportional relationship with few outliers.

**Figure 8. Plotting the Number of Requirements Expressed vs. the Number of Conflicts Detected in a Simulated Profile**

## Conclusions

In this paper, we recounted our methodology for performing rapid recertification tasks using formal analysis with our Application Profile Language. Our approach to automated

conflict detection was detailed. Based on this methodology, we identified three conflict reconciliation strategies that may be employed to resolve detected conflicts, and illustrated the process with a running example.

In our future work, we plan to extend the automation in our tool to provide automated recommendations to analysts for employing these conflict reconciliation strategies on existing profiles. We also plan to perform further performance analysis in order to objectively characterize the time savings gained by using this tool versus manual processes, and emerging formal analysis methods.

## References


Baader, F., Horrocks, I., & Sattler, U. (2005). Description logics as ontology languages for the semantic web. In F. Baader, I. Horrocks, & U. Sattler, *Mechanizing mathematical reasoning* (pp. 228–248). Berlin, Germany: Springer.

Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F., & Stein, L. A. (2004, February 10). *OWL Web Ontology Language.* Retrieved from W3C Reccomendation website: http://www.w3.org/TR/owl-ref/

Bell, D. E. (2005). Looking back at the Bell-LaPadula model. *Proceedings of the 21st Annual Computer Security Applications Conference*, 337–351.

Breaux, T., Anton, A., & Doyle, J. (2008). Semantic parameterization: A process for modeling domain descriptions*. ACM Transactions on Software Engineering Methodology*, 1–44.

Breaux, T. D., Hibshi, H., & Rao, A. (2014). Eddy, a formal language for specifying and analyzing data flow specifications for conflicting privacy requirements. *Requirements Engineering*, 281–307.

Brooks, F. P. (1995). "No silver bullet" refired. In *The Mythical Man Month*. Retrieved from Addison-Wesley.

Landwehr, C. (1981). *Formal models for computer security*. New York, NY: Association for Computing Machinery.

Lynxworks. (2007). *GE Fanuc Embedded Systems selected by Raytheon for Zumwalt Class Destroyer Program*. Retrieved from http://www.businesswire.com/news/home/20070725005359/en/GE-Fanuc-Embedded-Systems-Selected-Raytheon-Zumwalt#.VTACepPGqEw

O'Neil, W. D. (2007). *The Cooperative Engagement Capability (CEC).* Center for Technology and National Security Policy.

O'Rourke, R. (2012). *Navy DDG-51 and DDG-1000 Destroyer programs: Background and issues for Congress*. Washington, DC: Congressional Research Service.

Tolley, A. L., & Ball, J. E. (2014). Dual-Band Radar development: From engineering design to production. *NAVSEA Leading Edge, 7*(2), 53–61.


ACQUISITION RESEARCH PROGRAM
GRADUATE SCHOOL OF BUSINESS & PUBLIC POLICY
NAVAL POSTGRADUATE SCHOOL
555 DYER ROAD, INGERSOLL HALL
MONTEREY, CA 93943

www.acquisitionresearch.net